Project Title: Online Language Processing Tools

Applicant: Axel T. Schreiner  Telephone: 5-4902  e-mail: ats@cs.rit.edu
Department: Computer Science  College: GCCIS

Summary: Last summer, on my own time, I wrote tools to improve our Language Processing course 4003-580. The principal outcome was a website <http://www.cs.rit.edu/~ats/projects/lp/doc/> where students can paste or type compiler front-ends and back-ends into text areas and submit them to a web server; a servlet compiles the code and returns an applet; the students can then execute the result (their compiler) on a web page. I have used the tutorial already as the central part of the undergraduate Language Processors and the graduate Compiler Construction courses, 4003-580 and 4005-711, and for a quick introduction in the Programming Language Concepts course 4003-450. The very interactive nature of this website significantly improved my classroom presentation and visibly strengthened student motivation. Without any prompting, several student teams incorporated similar interactive facilities, based on my framework, into their course project presentations.

I propose to extend the site, e.g., to cover memory allocation algorithms, and to start other tutorials which apply the technology to Java instruction and operating system simulation. The servlets, tools, and interactive tutorial on language processing will be improved and expanded. The tutorial will be used as an instructional module in several different undergraduate and graduate computer science courses, see below. New tutorial materials specifically based on the Java compilation service <http://www.cs.rit.edu/~ats/projects/lp/doc/js/package-summary.html> will be produced as instructional modules and used at least in the Operating Systems 1 and Advanced Programming courses, 4003-440 and 4003-707.

Targeted Learners:

Compiler construction is a classic discipline within computer science. We currently offer two courses, both meet four hours per week:

• 4003-580 Language Processors, elective in Computer Science, suggested for the 4th and 5th year.

• 4005-711 Compiler Construction, required in the Languages and Tools cluster for graduate students in Computer Science.

However, the skills required to program compilers are applicable to a broader problem set. Therefore, some amount of instruction on language processing is part of a number of other courses:
• 4003-450 Programming Language Concepts in Computer Science, Software Engineering, and Computer Engineering Technology, required, suggested for the third year. About 10 of 40 hours deal with language processing.

• 4003-451 XML — Architecture, Tools, Techniques, elective in Computer Science, suggested for the third year and above. About 8 of 40 hours deal with conventional language processing.

• 4005-710 Programming Language Theory, elective for graduate students in Computer Science. If the course is based on the book *Programming Language Essentials* by Friedman et al. about 5 of 40 hours should deal with typing, etc. as implemented in Java.

The Java compilation service allows the creation of interactive tutorials to complement introductory Java instruction. This can affect students in the introductory programming sequence (i.e., CS1 and related courses) and in the Advanced Programming course 4003-707 for incoming graduate students.

<http://www.cs.rit.edu/~ats/projects/os1/doc/> details steps in implementing an operating system simulation for 4003-440 Operating Systems 1, a required course in Computer Science. The simulation should be interactive, based on the Java compilation service.

The proposed budget is for strengthening and expanding the existing framework and for developing more tutorial materials. Deployment to more courses necessitates an application server with a capacity appropriate for the number of users involved; according to our system administrators this is available. Deployment does not cause other costs.

The materials are an alternative to existing instruction. It is not expected that each and every section of the affected courses adopt these materials.

**Anticipated Impact:**

The conventional approach to language processing instruction is that students write toy compilers and other programs on a department-maintained workstation which provides a standard environment with the necessary compiler construction tools. Most students at that level own personal computers and prefer to work outside the lab; therefore, they have to install and maintain the environment on their own — the instructor's programming examples may or may not work there.

Experience with the existing tutorial confirms that vast improvements are possible. Server-based compilation eliminates the need to install tools elsewhere, browser-based editing and execution provides a reliable platform for experiments and demonstrations almost anywhere. Interaction with an instructor-provided example is possible simply by
pressing a button on a web page; students can edit the example right on the web page. As a result, language processing instruction allows significantly more what if experiments, in class or for self-study.

Similar effects are expected if Java or operating system tutorials are based on the same technology. E.g., rather then discussing a sorting technique — or at least executing a canned example — a web-based tutorial will incorporate code that students may revise, instrument, test, and improve.

In short, the technology enables much more learning by doing in a controlled but ubiquitous environment — through live demonstrations in the classroom or in blended learning.

**Impact on Student Success:**

The language processing tools and tutorial, as well as an operating system tutorial, target students who have already progressed beyond the initial hurdles in computer science — calculus and the introductory programming sequence. Experiments should improve understanding and make the subject more exciting, but I do not expect dramatic improvements in retention.

The Java tutorial modules are targeted at beginning students. Removing any handling problems from the interaction with programs should especially help weaker students to gain confidence, e.g., by "nudging" instructor-provided code. Therefore, if the tutorial modules are leveraged properly I would expect them to improve retention.

**Evaluation:** Evaluation will be within the courses I am scheduled to teach during the next academic year: 4005-710 Programming Language Theory in fall, 4003-707 Advanced Programming, 4005-711 Compiler Construction, 4003-580 Language Processors, and 4003-450 Programming Language Concepts in winter, and 4003-440 Operating Systems 1 and 4003-451 XML in spring. I will add suitable questions to the standard course evaluation for these courses to investigate the students' view of the different mode of instruction. The course project and homeworks will show how well the students can deal with the tools.

The tutorial allows adding compact and reasonably comprehensive language processing instruction to the Programming Language Concepts and the XML course, 4003-450 and 4003-451 — I would consider it a success if students in those courses can leverage the compiler tools to implement simple languages (which they cannot at present).

My work on the compiler tools seems to have a following outside of RIT. The tools can be downloaded from my web site and I get frequent requests for help with the tools. The interactive tutorial should make this more interesting for outsiders, too. To assess outside interest, I will ask
the computer science system administration staff to monitor the activity of the web site.

Thus far, my colleagues Anderson, Bischof, Carithers, Kwon, and Raj have expressed interest in using the materials. I am still discussing this with other members of the department. Their feedback will be part of my evaluation.

**Report:**

The materials, as well as my course notes, are always available on the Web and I have had numerous e-mail contacts from many countries about their use.

I have given a talk on the current state of the language processing tools [http://www.cs.rit.edu/~ats/talks/lp/ lp.mov](http://www.cs.rit.edu/~ats/talks/lp/lp.mov); I would produce a similar talk, at least within the colloquium series in our department, on any extensions. Other well-targeted forums could include O'Reilly’s online article series or the SIGCSE and SIGPLAN publications.

**Rationale:**

a. Last summer I built a preprocessor \( p_j \) to simplify using Java versions of the classic tools \( \text{lex} \) and \( \text{yacc} \) in the Language Processors course 4003-580. The ability to deliver all of this as a web service and, in particular, to execute the result in an applet without specifically programming for that came as a totally unexpected result — clearly, this is not regular college business.

b. The relevance of the materials to instruction in various courses has been discussed above.

c. As discussed, I have presented the tools to other computer science faculty; I think the materials are self-explanatory enough to be used without extensive prior instruction.

d. I have the necessary experience: I have worked on compiler tools, programming instruction, and (to a lesser degree) operating systems throughout my professional life.

e. The innovative aspects have been discussed above.

**Timetable:**

If this proposal is approved I will spend June through August 2005 improving the tools and the language processing and operating systems tutorials. I will hire and supervise a student (Dustin Mulcahey) to create Java tutorial materials during the same time.

As outlined above, I would use the materials in the courses I am scheduled to teach next year. I am also urging colleagues to use the materials in their sections of some of these courses.

Evaluations and other feedback can be analyzed following each quarter. However, the best time to present a complete summary at a computer science colloquium is early in fall 2006.